TRADING BOT

Automation

Bilal Shafique

Source Code:

```
import undetected_chromedriver as undetected_chrome
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
import random

# user details
user_email_address = input("Enter your email address: ")
user_password = input("Enter your password: ")
desired_user_id = 41641741

is_first_trade = True

# Taking input from the user about the desired profit
desired_profit_amount = float(input("Enter your profit amount you need in Dollars: "))

driver = undetected_chrome.Chrome()
driver.implicitly_wait(15)
driver.get("https://qxbroker.com/en")
driver.maximize_window()

# Function to press the Up Button
def pressing_up_button():
    """pressing the Up Button"""
    up_button_element = driver.find_element(
        By.XPATH, '//div[@class="section-deal__success  percent"]')
```

```
    up_button_element.click()


# Function to press the Down Button
def pressing_down_button():
    """pressing the Down Button"""
    down_button_element = driver.find_element(
        By.XPATH, "//button[@class='button button--danger button--spaced put-btn section-deal__button
']")
    down_button_element.click()


# Function to set the initial trading amount
def set_initial_trade_amount(initial_amount):
    """Sets the initial value for the first trade"""
    investment_section_element = driver.find_element(
        By.XPATH, '//*[@id="root"]/div/div[1]/main/div[2]/div[1]/div/div[5]/div[2]/div/div/input')
    investment_section_element.click()
    investment_section_element.send_keys(Keys.CONTROL + 'a')
    investment_section_element.send_keys(Keys.BACK_SPACE)
    investment_section_element.send_keys(initial_amount)


# Function to update the trading amount
def update_trade_amount(new_amount):
    """Updates the value for the next trades"""
    investment_section_element = driver.find_element(
        By.XPATH, '//*[@id="root"]/div/div[1]/main/div[2]/div[1]/div/div[5]/div[2]/div/div/input')
    investment_section_element.click()
    investment_section_element.send_keys(Keys.CONTROL + 'a')
    investment_section_element.send_keys(Keys.BACK_SPACE)
    investment_section_element.send_keys(new_amount)
```

```
# Function to retrieve the current time
def get_current_time():
    """Returns the seconds of the current minute"""
    timer_element = driver.find_element(By.XPATH, '//div[@class="server-time online"]')
    time_components = timer_element.text.split()
    current_time = time_components[0]
    seconds_part = current_time[-2:]
    return seconds_part

# Starting of the Code

# Clicking on the Login Button
login_button_element = driver.find_element(By.XPATH, "//*[@id='top']/div/div[1]/a[2]")
login_button_element.click()

# Entering Email and Password then Clicking on the Sign-in Button
email_entry_element = driver.find_element(
    By.XPATH, "//*[@id='tab-1']/form/div[1]/input").send_keys(user_email_address)
password_entry_element = driver.find_element(
    By.XPATH, "//*[@id='tab-1']/form/div[2]/input").send_keys(user_password)
signin_button_element = driver.find_element(
    By.XPATH, "//*[@id='tab-1']/form/button/div").click()

# Providing time to handle any verification code sent through email

# Selecting Demo Account from User Menu
time.sleep(10)
user_menu_element = driver.find_element(
```

```
    By.XPATH, "//div[@class='usermenu__info-wrapper']")
user_menu_element.click()

# Verifying User ID
user_id_element = driver.find_element(
    By.XPATH, "//*[@id='root']/div/div[1]/header/div[8]/div[2]/div[2]/ul[1]/li[1]/div[2]/div/span")
user_id_text = user_id_element.text
user_id_components = user_id_text.split()
real_user_id = int(user_id_components[1])

# Selecting Demo Account By Matching the User-ID With the Given ID
try:
    if real_user_id == desired_user_id:
        demo_account_menu_element = driver.find_element(
            By.XPATH, "//*[@id='root']/div/div[1]/header/div[8]/div[2]/div[2]/ul[1]/li[3]/a")
        demo_account_menu_element.click()
        close_button_element = driver.find_element(
            By.XPATH, "//*[@id='root']/div/div[3]/div/div/div/div[2]/button").click()

        time.sleep(15)
        current_profit_amount = 0

        initial_trading_amount = int(input("Set your initial trading amount here: $"))
        current_trading_amount = initial_trading_amount
        set_initial_trade_amount(initial_trading_amount)

        demo_account_balance_element = driver.find_element(
            By.XPATH, "//*[@id='root']/div/div[1]/header/div[8]/div[2]/div/div[3]/div[2]")
        balance_text = demo_account_balance_element.text
```

```
previous_demo_account_balance = float(balance_text[1:].replace(',', ''))

buttons_list = [pressing_up_button, pressing_down_button]
current_button = random.choice(buttons_list)

while True:
    seconds = get_current_time()
    current_second = int(seconds)

    if current_second == 0:
        if is_first_trade:
            current_button = random.choice(buttons_list)
            current_button()
            is_first_trade = False
            time.sleep(2)
        else:
            print(previous_demo_account_balance)
            demo_account_balance_element = driver.find_element(
                By.XPATH, "//*[@id='root']/div/div[1]/header/div[8]/div[2]/div/div[3]/div[2]")
            balance_text = demo_account_balance_element.text
            current_demo_account_balance = float(balance_text[1:].replace(',', ''))
            print(current_demo_account_balance)
            profit_amount = current_demo_account_balance - previous_demo_account_balance
            remaining_profit = profit_amount

            current_profit_amount += int(profit_amount)

            print(f"Profit after trade: ${profit_amount}")
            print(f"Total Profit: ${current_profit_amount}")
```

```
                previous_demo_account_balance = current_demo_account_balance


                time.sleep(6)


                if current_profit_amount >= desired_profit_amount:
                    print(f"Desired profit of ${desired_profit_amount} achieved.")
                    driver.quit()
                    break


                if remaining_profit >= 0:
                    time.sleep(0.5)
                    current_trading_amount = initial_trading_amount
                    update_trade_amount(current_trading_amount)
                    current_button()


                else:
                    time.sleep(2)
                    current_trading_amount = str(
                        int(current_trading_amount) * 2)
                    current_button = pressing_down_button if current_button == pressing_up_button else
pressing_up_button
                    update_trade_amount(current_trading_amount)
                    current_button()


                remaining_profit = 0


except:
    print("An error occurred please look into your code")
```

Read me:

Project Title: Automated Trading Script for QXBroker Website

Description: Developed a Python script using Selenium for automating trading activities on the QXBroker website. The script allows users to log in, set profit targets, and execute trades based on predefined conditions. It employs randomization for trade direction and dynamically adjusts trade amounts based on previous outcomes.

Key Features:

- Seamless login process with user-provided credentials.
- Flexible profit targeting mechanism allowing users to specify desired profit amounts.
- Dynamic trade execution based on random direction selection and previous trade outcomes.
- Real-time monitoring of account balance and automatic trade adjustments.

Technologies Used: Python, Selenium

Achievement: Successfully implemented an automated trading solution, demonstrating proficiency in Python scripting and web automation techniques.